1. 上课约定须知

课程主题：全域离线数仓--day01--企业级电商数仓架构设计
上课时间：20:00 - 23:00
课件休息：21:30 左右 休息10分钟
课前签到：如果能听见音乐，能看到画面，请在直播间扣 666 签到

2. 本次内容大纲

今天的课程是 全域离线数仓的 第一次。第一次课程的内容，旨在帮助大家理解，为什么要构建数仓，构建数仓的流程，怎么构建数仓等等相关问题。

```
1、数据仓库概述
2、数仓构建流程
3、数仓建模理论
4、用户行为分析离线数仓案例
5、未来数据仓库发展
```

今天的课程的最大目的，是希望大家对于构建企业级数仓有一个全面的体系化认知。下两次课是具体实践。

3. 课程详细内容

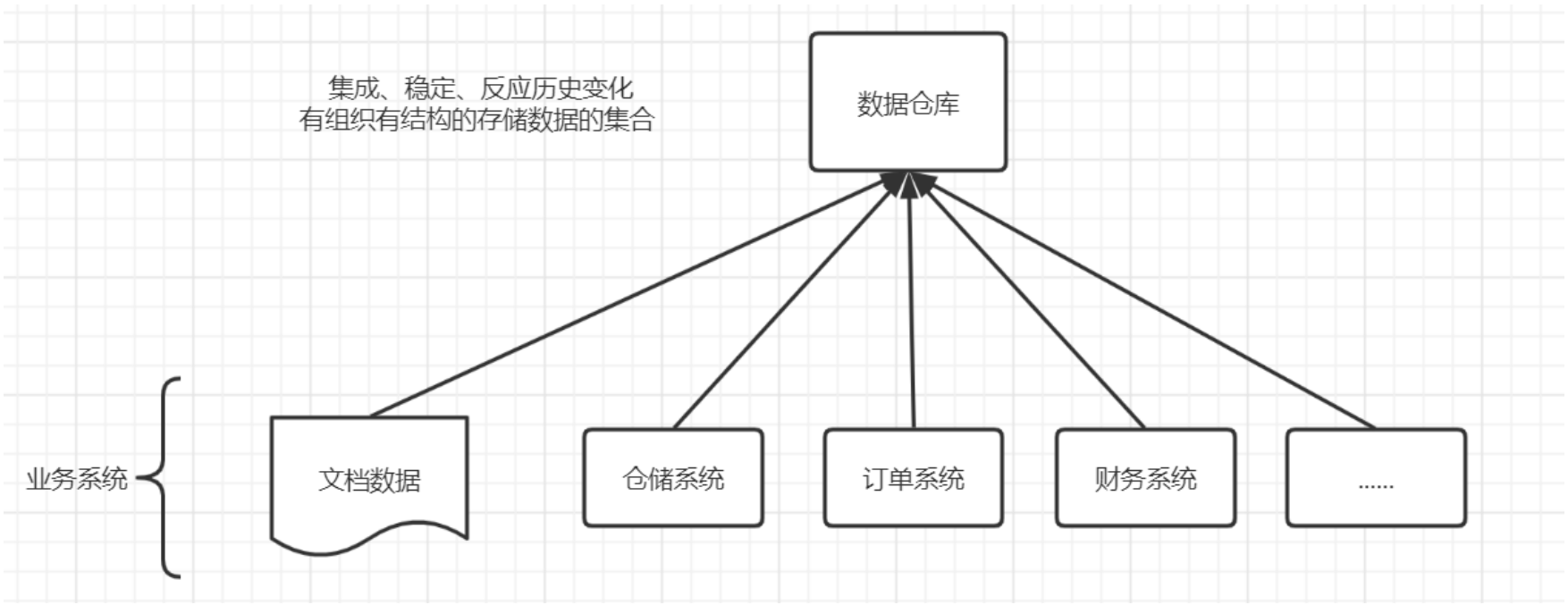
3.1. 认识数据仓库

3.1.1. 数据仓库概念

数据仓库，英文名称为 Data Warehouse，可简写为 DW 或 DWH。数据仓库，是为企业所有级别的决策制定过程，提供所有类型数据支持的战略集合。它是单个数据存储，出于分析性报告和决策支持目的而创建。 为需要业务智能的企业，提供指导业务流程改进、监视时间、成本、质量以及控制。

数据仓库之父比尔·恩门（Bill Inmon）在1991年出版的 “Building the Data Warehouse”（《建立数据仓库》）一书中所提出的定义被广泛接受--**数据仓库（Data Warehouse）是一个面向主题的（Subject Oriented）、集成的（Integrated）、相对稳定的（Non-Volatile）、反映历史变化（Time Variant）的数据集合，用于支持管理决策(Decision Making Support)。**

1、面向主题：在较高层次上将企业信息系统的数据综合归并进行分析利用的抽象的概念。每个主题基本上对应一个相应的分析领域。
2、集成的：企业级数据，同时数据要保持一致性、完整性、有效性、精确性
3、稳定的：从某个时间段来看是保持不变的，没有更新操作、删除操作，以查询分析为主
4、反应历史变化：数仓中的数据主要存储的是反应历史某个时刻的状态信息的数据



3.1.2. 传统数据仓库发展历程

1、**萌芽阶段**：数据仓库概念最早可追溯到 20 世纪 70 年代，MIT提出希望提供一种架构将业务处理系统和分析处理分为不同的层次。

2、**探索阶段**：20 世纪 80 年代，建立 Technical Architecture2 规范，该明确定义了分析系统的四个组成部分：数据获取、数据访问、目录、用户服务。

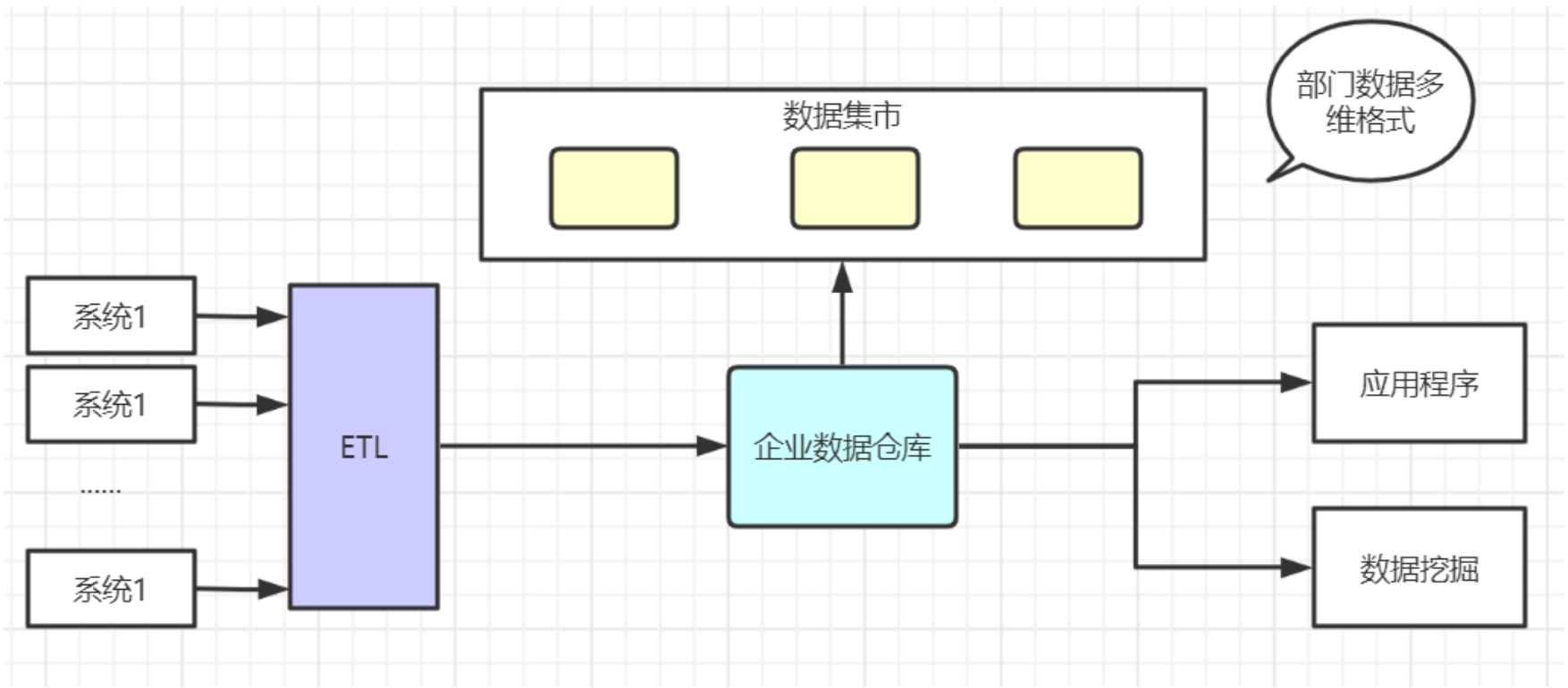
3、**雏形阶段**：1988年，IBM第一次提出信息仓库的概念：一个结构化的环境，能支持最终用户管理其全部的业务，并支持信息技术部门保证数据质量；抽象出基本组件：数据抽取、转换、有效性验证、加载、cube开发等，基本明确了数据仓库的基本原理、框架结构，以及分析系统的主要原则。

4、**确立阶段**：1991年，Bill Inmon出版《Building the Data Warehouse》提出了更具体的数据仓库原则：面向主题的、集成的、包含历史的、不可更新的、面向决策支持的、面向全企业的、最明细的数据存储、快照式的数据获取。尽管有些理论目前仍有争议，但凭借此书获得数据仓库之父的殊荣。

3.1.3. Bill Inmon 数仓

1991年，Bill Inmon 出版《Building the Data Warehouse》提出了更具体的数据仓库原则：数据仓库是面向主题的，集成的，包含历史的，不可更新的，面向决策支持的，面向全企业的，最明细的数据存储，数据快照式的数据获取，Bill Inmon 凭借此书获得 “数据仓库之父” 的称号

Bill Inmon 主张自上而下的建设企业数据仓库，认为数据仓库是一个整体的商业智能系统的一部分。一家企业只有一个数据仓库，数据集市的信息来源出自数据仓库，在数据仓库中，信息存储符合第三范式，大致架构：自上而下：分散异构的数据源 -> 数据仓库 -> 数据集市



1、操作型系统的数据和体系外数据需要经过 ETL 过程，加载到企业数据仓库中

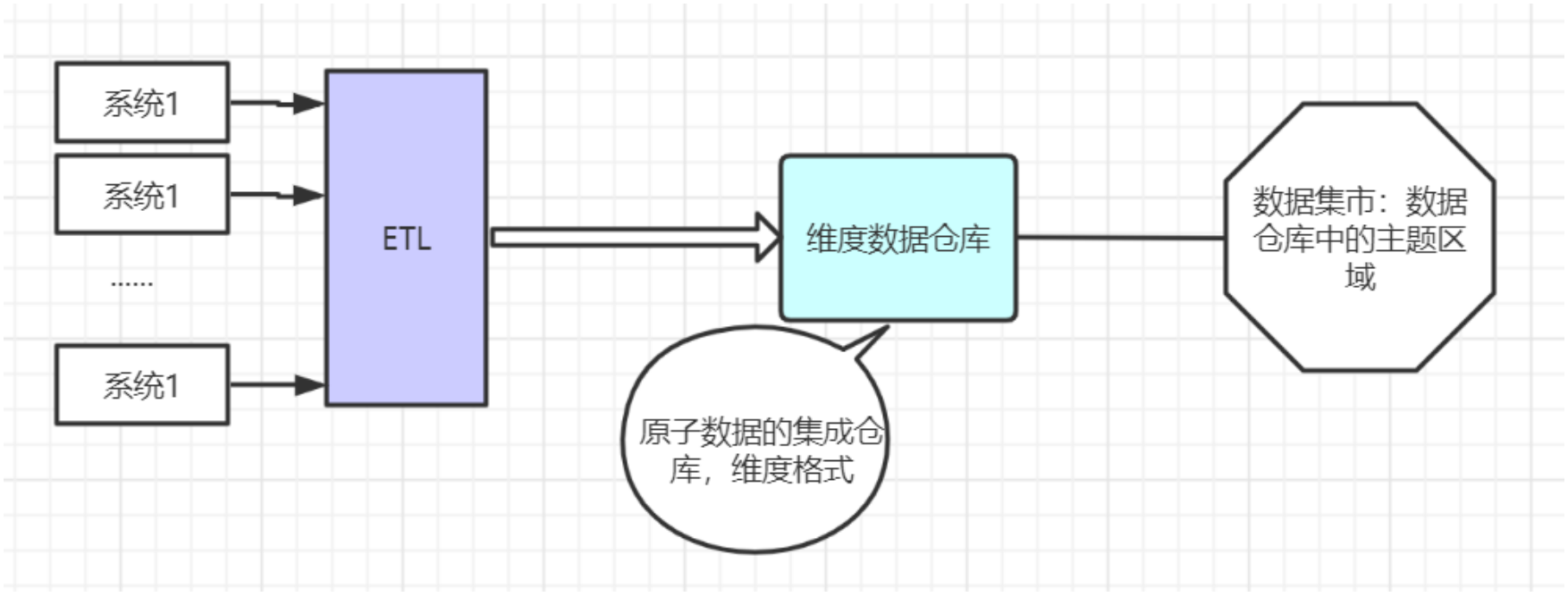
2、企业数据仓库是企业信息化工厂的枢纽，是原子数据的集成仓库，其目的是将附加的数据存储用于各类分析型系统；在数据仓库中会对数据进行清洗，并抽取实体-关系。

3、数据集市是针对不同主题的聚集区域

3.1.4. Ralph Kimball 数仓

Ralph Kimball 出版《The Data Warehouse Toolkit》，其主张自下而上的建立数据仓库，推崇建立数据集市，认为数据仓库是企业内所有数据集市的集合，信息总是被存储在多维模型当中，其思路：

```
Kimball 的模型是自底向上的，即从数据集市-> 数据仓库 -> 分散异构的数据源。
```



1、Kimball 的模型的数据源往往是给定的若干个数据库表，数据较为稳定但是数据之间的关联关系比较复杂，需要从这些 OLTP 中产生的事务型数据结构抽取出分析型数据结构。Kimball 是以最终任务为导向，将数据按照目标拆分出不同的表需求，通过 ETL 导入数据集市层

2、Kimball 模型将分散异构的数据源经 ETL 转化为事实表和维度表导入数据集市，数据集市由若干个事实表和维度表组成

3、在数据集市将事实表和维度表根据分析主题组合后导入数据仓库中，用于数据分析

3.1.5. Inmon与Kimball 建模总结

两种思路和观点在实际的操作中都很难成功的完成项目交付，直至最终 1998 年 Bill Inmon 提出了新的 BI 架构 CIF(Corporation Information Factory，企业信息工厂)，把数据集市包含了进来。

CIF 的核心是将数仓架构划分为不同的层次以满足不同场景的需求，比如常见的 ODS（Operational Data Store）、DW（Data Warehouse）、DM（Data Market）等，每层根据实际场景采用不同的建设方案，该思路也是目前数据仓库建设的架构指南，但自上而下还是自下而上的进行数据仓库建设，并未统一，并不是绝对的。

第一：共同点

```
1、均极力推崇数据仓库，认为从 OLTP 到 BI 分析之间建立数据仓库是很有必要的；
2、均认为数据仓库的建立需要从企业整体角度出发，迭代开发，尽量避免按部门建立独立的数据仓库；
3、数据进入数据仓库之前，需要经过 ETL 整合。
```

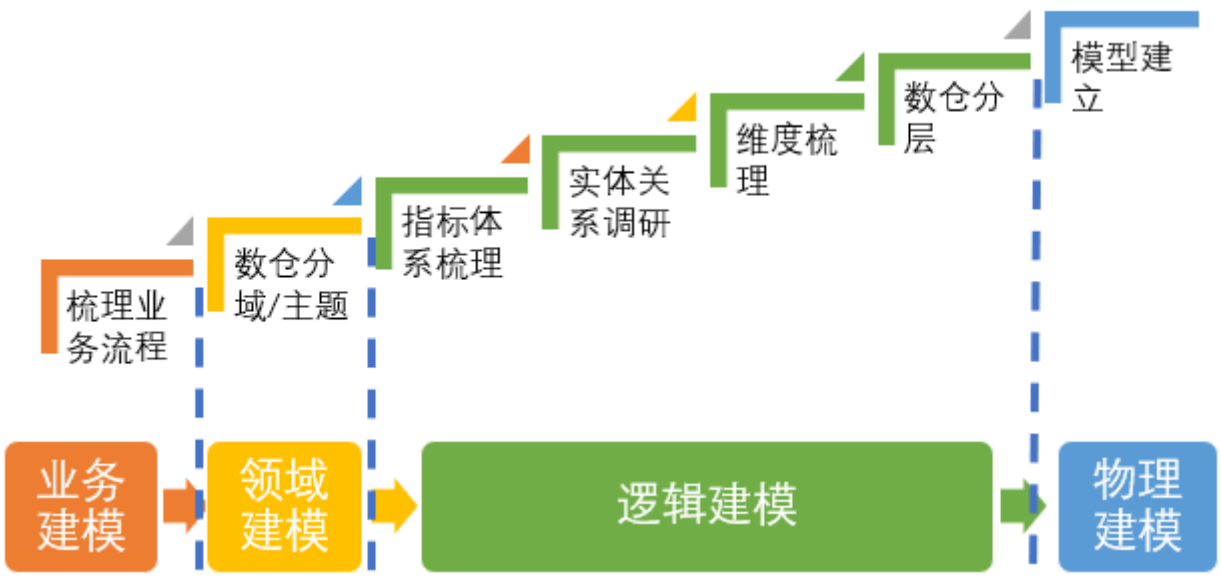
第二：不同点

```
Inmon 理论：构建数仓
1、数仓概念：数据仓库（Data Warehouse）是一个面向主题的（Subject Oriented）、集成的（Integrated）、相对稳定的（Non-Volatile）、反映历史变化（Time Variant）的数据集合，用于支持管理决策（Decision Making Support）
2、自上而下按照主题建立数据仓库，如按照客户、供应商、产品等建立不同的主题，开发过程中每次增加一个主题
3、当建立的数据集市是跨多个主题的，需要以整合好的主题数据为基础。

Kimball 理论：维度建模
1、自下而上，维度建模
2、先按照业务主线建立最小粒度的事实表，再建立维度表，形成数据集市，通过"一致维度"能够共同看到不同数据集市的信息；
```

3.2. 数仓构建流程

吐槽一下：绝大多数互联网公司，几乎没有建模，也没有文档，上去就是干，最后就是一堆的宽表。

数仓建模的过程分为业务建模、领域建模、逻辑建模和物理建模，但是这些步骤比较抽象。为了便于落地，我根据自己的经验，总结出上面的七个步骤：梳理业务流程、垂直切分、指标体系梳理、实体关系调研、维度梳理、数仓分层以及物理模型建立。



业务流程

1、找到公司核心业务流程，找到谁，在什么环节，做什么关键动作，得到什么结果。
2、梳理每个业务节点的客户及关注重点，找到数据在哪。

数仓分域/主题

1、决定数仓的建设方式，快速交活，就用自下而上的建设。要全面支撑，就顶层规划，分步实施，交活稍微慢点。
2、同时按照业务领域划分主题域。主题域的划分方法有：按业务流划分（推荐）、按需求分、按职责分、按产品功能分等。

指标体系

1、指标的意义在于统一语言，统一口径（`user,username`）。所以指标的定义必须有严格的标准。否则如无根之水。指标可分为原子指标、派生指标和衍生指标。
2、依照指标体系建设标准，开始梳理指标体系。整个体系同样要以业务为核心进行梳理。同时梳理每个业务过程所需的维度。维度就是你观察这个业务的角度，指标就是衡量这个业务结果 好坏的量化结果。

实体关系

1、每个业务动作都会有数据产生。我们将能够获取到的数据，提取实体，绘制ER图，便于之后的维度建模。
2、同样以业务过程为起点向下梳理，此时的核心是业务表。把每张表中涉及的维度、指标都整理出来。

维度整理

1、维度标准化是将各个业务系统中相同的维度进行统一的过程。其字段名称、代码、名字都可能不一样，我们需要完全掌握，并标准化。
2、维度的标准尽可能参照国家标准、行业标准。例如地区可以参照国家行政区域代码。另外，有些维度存在层级，如区域的省、市、县。绝大多数业务系统中的级联就是多层级维度。

数仓分层

1、数据仓库一般分为4层，名字可能会不一样，但是其目的和建设方法基本一致，每一层采用的建模方法都不一样，其核心是逐层解耦。越到底层，越接近业务发生的记录，越到上层，越接近业务目标。
2、依托数仓分层的设计理论，根据实际业务场景，我们就可以梳理出整体的数据流向图。这张图会很清晰的告诉所有人，数据从那来，到哪里去，最终提供什么样的服务。

模型建立

1、此时才真正进入纯代码阶段。数仓、ETL工具选型；ETL流程开发；cube的建立；任务调度，设定更新方式、更新频率；每日查看日志、监控etl执行情况等等。

干货经验

1、数仓建设必须从业务中来，到业务中去；
2、数仓分层的目的是业务解耦；
3、无论哪种建模方式，其核心是业务实体；
4、按领域建设能快速交活，后遗症将会在2年之后爆发，且难以解决；
5、数仓建设应该把75%的时间投入到设计阶段，如果不是，那你就惨了；
6、数仓本身也可以迭代。
7、传统数仓并没有一种叫做“宽表模型”的模型，大数据时代新诞生的名词，因为很多大数据组件 `join` 代价极高。实际上是范式退化。

3.3. 数仓建模基本理论

3.3.1. 建模目标

数据模型就是数据组织和存储方法，它强调从业务、数据存取和使用角度合理存储数据。Linux 的创始人 Torvalds 有 一段关于 "什么才是优秀程序员" 的话：“烂程序员关心的是代码，好程序员关心的是数据结构和它们之间的关系”，其阐述了数据模型的重要性。有了适合业务和基础数据存储环境的模型，那么大数据就能获得以下好处。

```
1、访问性能：能够快速查询所需的数据，减少数据I/O
2、数据成本：减少不必要的数据冗余，实现计算结果数据复用，降低大数据系统中的存储成本和计算成本
3、使用效率：改善用户应用体验，提高使用数据的效率
4、数据质量：改善数据统计口径的不一致性，减少数据计算错误的可能性，提供高质量的、一致的数据访问平台
```

所以，大数据的数仓建模需要通过建模的方法更好的组织、存储数据，以便在性能、成本、效率和数据质量之间找到最佳平衡点。

核心：以空间换时间 以时间换空间

```
1、以空间换时间：Join = 维度表+事实表 ===> 大宽表
2、以时间换空间：本身一张大事实表 ==> 事实表 + 多张维度表
```

3.3.2. 关系型数据库范式

设计关系数据库时，遵从不同的规范要求，设计出合理的关系型数据库，这些不同的规范要求被称为不同的范式。

关系模式范式：关系型数据库设计时，遵照一定的规范要求，目的在于**降低数据的冗余性**和**提高数据的一致性**，目前业界范式有

```
第一范式(1NF)：字段不可分，每个字段是原子级别的，多个字段组织在一起形成一个字段是违反第一范式的，简单说，保证列的原子性
第二范式(2NF)：有主键，非主键字段依赖主键。简单说，保证行的原子性，每一行都有唯一的主键，其他字段的值与主键一一对应
第三范式(3NF)：非主键字段不能相互依赖，简单说，保证表的原子性，每张表中的数据不会冗余，一旦有冗余字段，就需要拆一张表出来，用外键与主表关联
巴斯-科德范式(BCNF)：在3NF基础上，任何非主字段不能对主键子集依赖
第四范式(4NF)：在满足3NF的基础之上，表中不能包含一个实体的两个或多个互相独立的多值因子
第五范式(5NF)完美范式：在满足4NF的基础之上，表必须可以分解为较小的表，除非那些表在逻辑上拥有与原始表相同的主键
```

各种范式呈递次规范，越高的范式数据库冗余越小。有冗余的数据库未必是最好的数据库， 有时为了提高运行效率，就必须降低范式标准，适当保留冗余数据。

满足最低要求的范式是第一范式（1NF）。在第一范式的基础上进一步满足更多规范要求的称为第二范式（2NF），其余范式以次类推。一个数据库设计如果符合第二范式，一定也符合第一范式。如果符合第三范式，一定也符合第二范式。一般说来，数据库只需满足第三范式(3NF）就行了。 总之，规范化的过程就是在数据库表设计时移除数据冗余的过程。随着规范化的进行，数据冗余越来越少，但数据库的效率也越来越低。

三大范式只是一般设计数据库的基本理念，可以建立冗余较小、结构合理的数据库。如果有特殊情况，当然要特殊对待，数据库设计最重要的是看需求跟性能，需求>性能>表结构。所以不能一味的去追求范式建立数据库。

3.3.3. ER 对象关系实体模型

```
1、数据库：存储了数据，专门做针对单条记录的 增删改查
2、数据仓库：存储了数据，专门用来做大量数据的分析 select 查询分析
```

在信息系统中（CMS, ERM, OA等），将事物抽象为“实体”、“属性”、“关系”来表示数据关联和事物描述；实体：Entity，关系：Relationship；这种对数据的抽象建模通常被称为 ER 实体关系模型。

实体：通常为参与到过程中的主体，客观存在的，比如商品、仓库、汽车。此实体非数据库的实体表

属性：对主体的描述、修饰即为属性，比如商品的属性有商品名称、颜色、尺寸、重量、产地等

关系：现实的物理事件是依附于实体的，比如商品入库事件，依附实体商品、货位，就会有“库存”的属性产生；用户购买商品，依附实体用 户、商品，就会有“购买数量”、“金额”的属性产品。

实体之间建立关系时，存在对照关系：

```
1:1     即1对1的关系，比如实体人、身份证，一个人有且仅有一个身份证号
1:n     即1对多的关系，比如实体学生、班级，对于某1个学生，仅属于1个班级，而在1个班级中，可以有多个学生
n:m     即多对多的关系，比如实体学生、课程，每个学生可以选修多门课程，同样每个课程也可以被多门学生选修
```

在日常建模过程中：所以 ER 实体关系模型也可以称作 E-R 关系图

```
“实体”用矩形表示
“关系”用菱形表示
“属性”用椭圆形表示
```

应用场景：

```
ER模型是数据库设计的理论基础，当前几乎所有的OLTP系统设计都采用ER模型建模的方式
Bill Inom提出的数仓理论，推荐采用ER关系模型进行建模
BI架构提出分层架构，数仓底层ods、dwd、dm也多采用ER关系模型就行设计
```

3.3.4. 维度模型

核心：事实表 + 维度表

下单业务：

```
1、事实表：某个用户，在某个平台的某个商家中，在某个时刻，买了某个商品。 一条事实数据
2、实体：用户，商品，商家....
3、维度：IP，时间，手机类型（小米，华为，oppo），....
```

维度建模从分析决策的需求出发构建模型，为分析需求服务，因此它重点关注用户如何更快速地完成需求分析，同时具有较好的大规模复杂查询的响应性能。其典型的代表是星形模型，以及在一些特殊场景下使用的雪花模型。其设计分为以下几个步骤：

1、选择需要进行分析决策的业务过程。业务过程可以是单个业务事件，比如交易的支付、退款等；也可以是某个事件的状态，比如当前的账户余额等；还可以是一系列相关业务事件组成的业务流 程，具体需要看我们分析的是某些事件发生情况，还是当前状态， 或是事件流转效率。

2、选择粒度。在事件分析中，我们要预判所有分析需要细分的程度，从而决定选择的粒度。粒度是维度的一个组合。

3、选择维度。选择好粒度之后，就需要基于此粒度设计维表，包括维度属性，用于分析时进行分组和筛选。

4、选择事实。确定分析需要衡量的指标。

Ralph Kimball 推崇数据集市的集合为数据仓库，同时也提出了对数据集市的维度建模，将数据仓库中的表划分为事实表、维度表两种类型。

3.3.5. 事实表

在 ER 模型中抽象出了有实体、关系、属性三种类别，在现实世界中，每一个操作型事件，基本都是发生在实体之间的，伴随着这种操作事件的发生，会产生可度量的值，而这个过程就产生了一个事实表，存储了每一个可度量的事件。

电商场景：一次购买事件，涉及主体包括客户、商品、商家，产生的可度量值，包括商品数量、金额、件数等，所以订单明细表，就是一张事实表。

- **原则1: 尽可能包含所有与业务过程相关的事实。**事实表设计的目的是为了度量业务过程，事实越多，越有利于多角度多维度度量业务
- **原则2: 只选择与业务过程相关的事实。**比如下单事件，不应该存储支付金额
- **原则3: 分解不可加性事实为可加的组件。**比如订单的优惠率，应该分解为订单的原价与订单优惠金额两个事实存储在事实表中
- **原则4: 在选择维度和事实之前必须先声明粒度。**粒度是维度的组合。先确定粒度，再确定维度。粒度用于确定事实表中一行所表示业务的细节层次，决定了维度模型的扩展性，在选择维度和事实之前必须先声明粒度，且每个维度和事实必须与所定义的粒度保持一致。
- **原则5: 在同一个事实表中不能有多种不同粒度的事实。**事实表中的所有事实需要与表定义的粒度保持一致，在同一个事实表中不能有多种不同粒度的事实。
- **原则6: 事实的单位要保持一致，对于同一个事实表中事实的单位应该保持一致。**比如订单的金额、订单优化金额、订单运费金额这三个事实，应该采用一致的计量单位，统一为元或者分，方便实用。
- **原则7: 对事实的 null 值要处理。**对于事实表中事实度量为null 值的处理，因为在数据库中null 值对常用数字型字段的sql过滤条件都不生效，比如大于，小于，等于...，建议用零值填充。
- **原则8: 使用退化维度提高事实表的易用性。**这样设计的主要目的是为了减少下游用户使用时关联多个表进行操作。直接通过退化维度实现事实表的操作。通过增加存储的冗余，提高计算的速度。空间置换时间的方式。

3.3.6. 维度表

维度，顾名思义，看待事物的角度。比如从颜色、尺寸的角度来比较手机的外观，从 cpu、内存等比较手机性能。

维度表一般为单一主键，在 ER 模型中，实体为客观存在的事物，会带有自己的描述性属性，属性一般为文本性、描述性的，这些描述被称为维度。

比如商品，单一主键：商品 ID，属性包括产地、颜色、材质、尺寸、单价等，但并非属性一定是文本，比如单价、尺寸，均为数值型描述性的，日常主要的维度抽象包括：时间维度表、地理区域维度表、类别维度等。

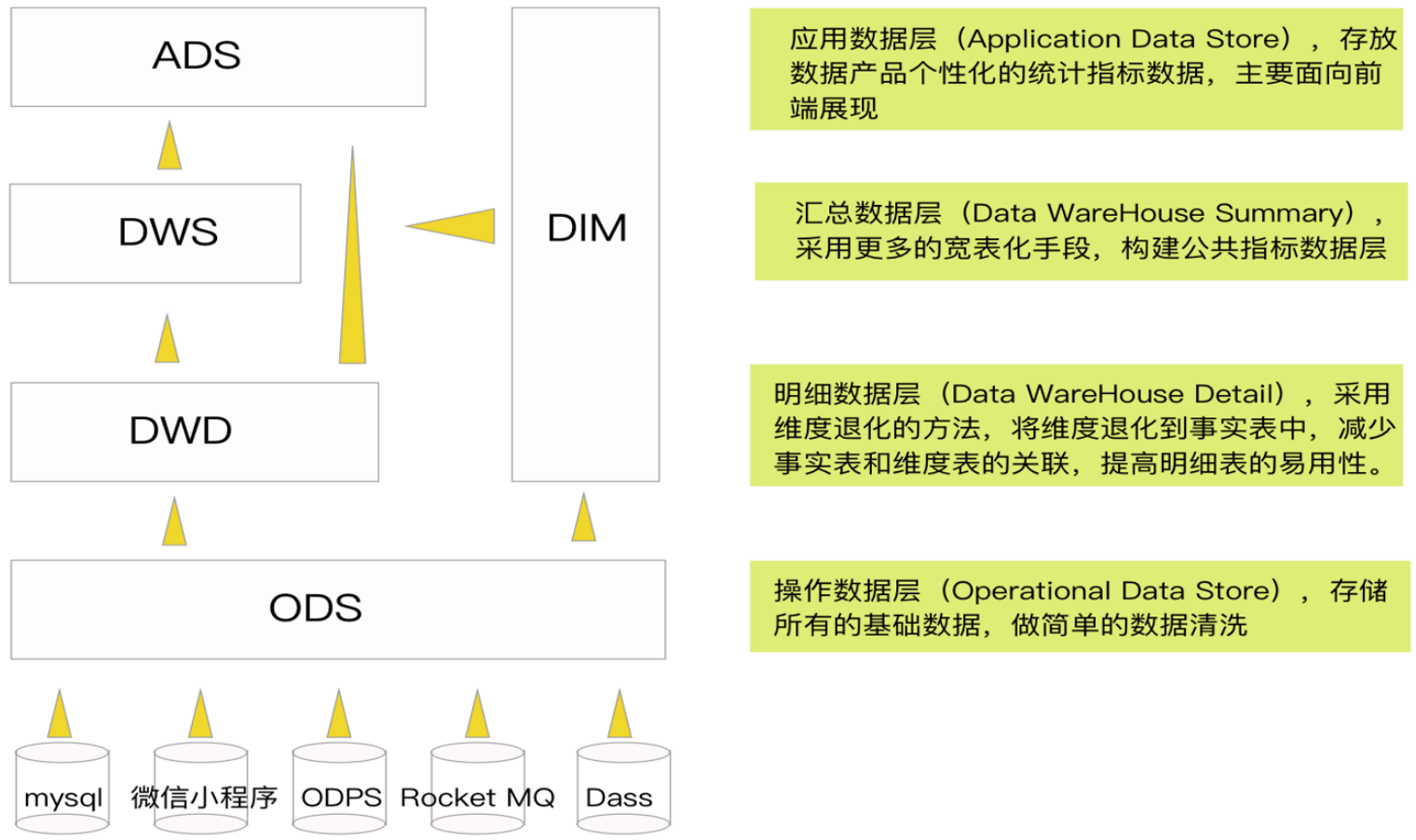
维度表设计原则：

- **原则1**：每个维表必须有而且只有一个最明细层作为该维表的颗粒度。
- **原则2**：任何一个维表若被多个事实表使用，尽量沉淀出通用的维度属性，这些属性作为公共维表属性来设计公共维表。
- **原则3**：除非出于性能考虑，否则每一个非键属性应只出现在一张维表里。
- **原则4**：尽可能生成丰富的维度属性，尽可能多地给出包括一些富有意义的文字性描述
- **原则5**：维表应尽量保存业务使用的代码和 ID ，以及描述信息。
- **原则6**：维表的主键（代理键）应做为事实表的外键包含在事实表内。
- **原则7**：每个维表中要有相应的行记录来处理特殊的情形来避免在事实表中置空值。如记录不存在，以及迟到的维记录。
- **原则8**：通常情况下，一个维度模型，不应该携带很多的维度，否则会增加查询的负担，响应性能，个位数最佳
- **原则9**：需要记录属性变化的维的主键应该是使用代理键，并使用具有业务含义，业务用户可识别的代码作为自然键。业务系统自带的代理键不能做为维表的主键。

3.3.7. 模型分层

分层通用做法：

分层名称	意义	解释	建模方法
ODS	原始数据层	Operational Data Store，操作数据层，即原始数据层，与业务系统基本同构（可能会增加管理字段），具的是保留历史,解耦业务数据库。	
DWD	明细数据层	Data Warehouse Detail，明细数据层，对数据进行清洗、代码统一、字段统一、格式统一等工作，目的是标准化,给后续处理提供干净、统一、标准的数据。	3NF模型
DWB	基础数据层	Data Warehouse Base，基础数据层，遵照维度模型的原理，将数据拆成维度和事实，进行维度、事实的统一。对数据进行轻度汇总，形成指标结果。	维度模型
DWS	汇总数据层	Data Warehouse Service，服务数据层，按照业务目标，DWS 对已经处理好的数据进行横向汇聚、纵向汇总。按照宽表模型进行数据冗余和预计算,以空间换时间。	宽表模型
ADS	数据应用层	Application Data Store，应用数据层，主要是提供给数据产品和数据分析使用的数据，一般会同步到在 MySQL 等系统中供线上系统使用，也可能会存在 Hive 或者 Druid 中供数据分析和数据挖掘使用。经常说的报表数据，或者说那种大宽表，一般就放在这里。	
DIM	公共维度层	Dimension，公共维度层，主要存放公共的信息数据，如国家代码和国家名，地理位置等信息就存在DIM层表中，对外开放，用于DWD，DWS和APP层的数据维度关联。	



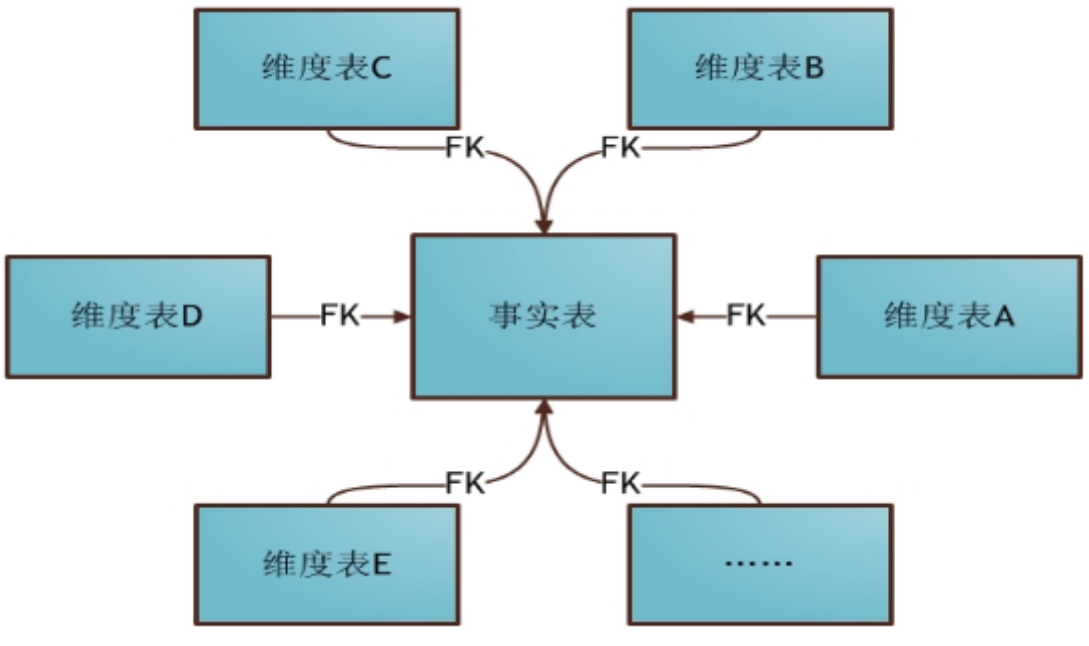
3.3.8. 模型分类

在构建数据仓库的维度建模中，一般有三种模式。**星型模型/雪花模型/星座模型**

在多维分析的商业智能解决方案中，根据事实表和维度表的关系，又可将常见的模型分为**星型模型**和**雪花型模型**。在设计逻辑型数据的模型的时候，就应考虑数据是按照星型模型还是雪花型模型进行组织。

3.3.9. 星型模型

当所有维表都直接连接到“ 事实表”上时，整个图解就像星星一样，故将该模型称为星型模型。 **星型架构是一种非正规化的结构，多维数据集的每一个维度都直接与事实表相连接，不存在渐变维度，所以数据有一定的冗余**



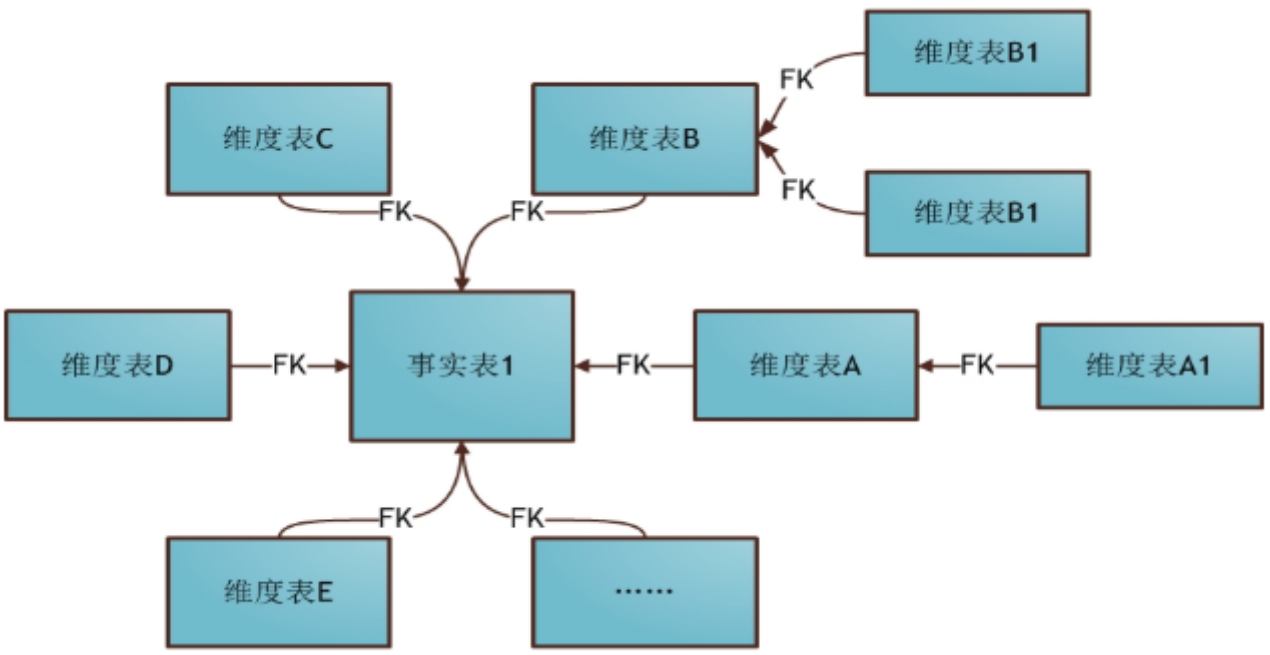
星形模型的维度建模由一个事实表和一组维度表组成，有以下特点：

1、维度表只和事实表关联，维度表之间没有关联；
2、每个维表的主码为单列，且该主码放置在事实表中，作为两边连接的外码；
3、以事实表为核心，维表围绕核心呈星形分布；

3.3.10. 雪花模型

当一个或多个维表没有直接连接到事实表上，而是通过其他维表连接到事实表上时，其图解就像多个雪花连接在一起，故称雪花模型。雪花模型是对星型模型的扩展。它对星型模型的维表进一步层次化，原有的各维表可能被扩展为小的事实表，形成一些局部的"层次"区域，这些被分解的表都连接到主维度表而不是事实表。

优点：**通过最大限度地减少数据存储量以及联合较小的维表来改善查询性能。**雪花型结构去除了数据冗余。

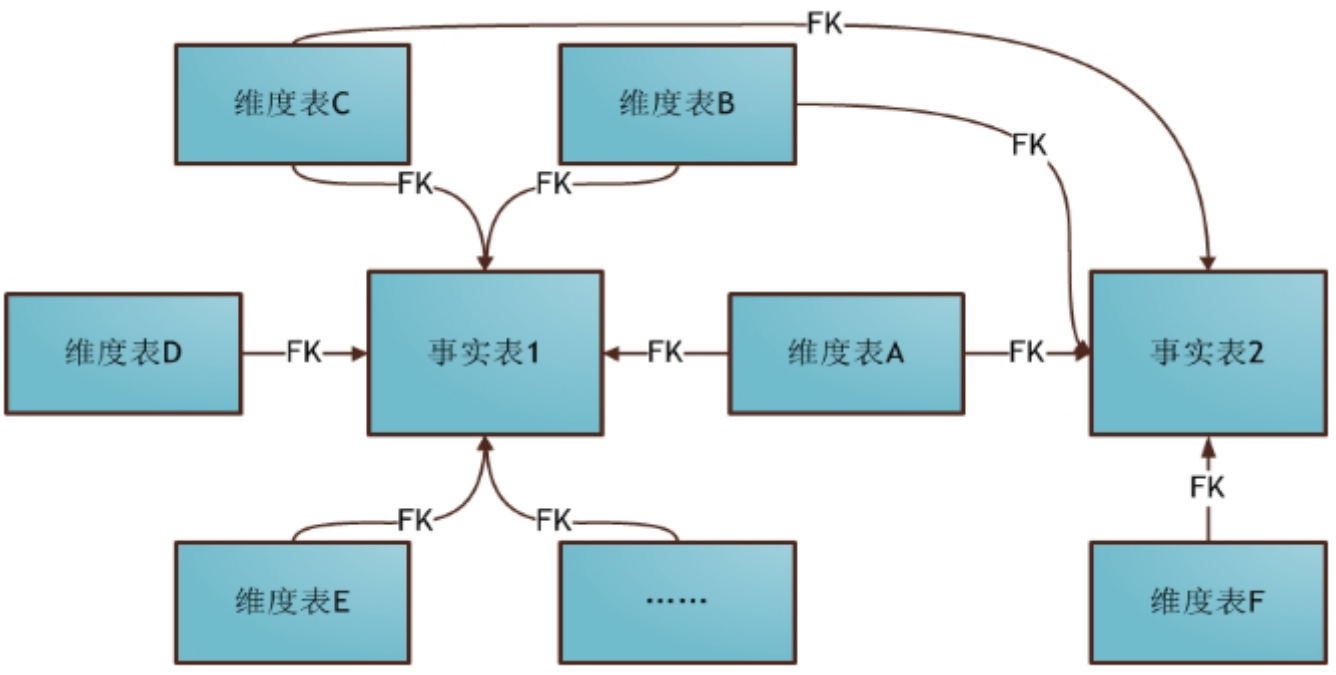


雪花模型的特点：

1、将星形模式的大维度表拆分未小维度表，满足规范化设计，但不利于开发。
2、星型模型因为数据的冗余所以很多统计查询不需要做外部的连接，因此一般情况下效率比雪花型模型要高。在冗余可以接受的前提下，实际运用中星型模型使用更多，也更有效率。
3、在雪花模型中，数据模型的业务层级是由一个不同维度表主键-外键的关系来代表的。而在星形模型中，所有必要的维度表在事实表中都只拥有外键。

3.3.11. 星座模型

星座模式也是星型模式的扩展。其实就是多个雪花模型连在一起。



星座模型特点：

1、星型模型和雪花模型都是基于多个维表对应事实表，有的时候一个维度表可能被多个事实表用到，这个时候就需要采用星座模式。

3.3.12. 模型对比

雪花模型是将星型模型的维度表进一步划分，使得各维度表满足规范化设计。而星座模型允许星型模型中出现多个事实表，更符合实际业务需求。 雪花模型使得维度分析更加容易。

综上所述可以看出：星型模型和雪花模型主要区别就是对维度表的拆分：

```
1、对于雪花模型，维度表的涉及更加规范，一般符合3NF；
2、而星型模型，一般采用降维的操作，利用冗余来避免模型过于复杂，提高易用性和分析效率
```

星型模型和雪花模型的主要区别：

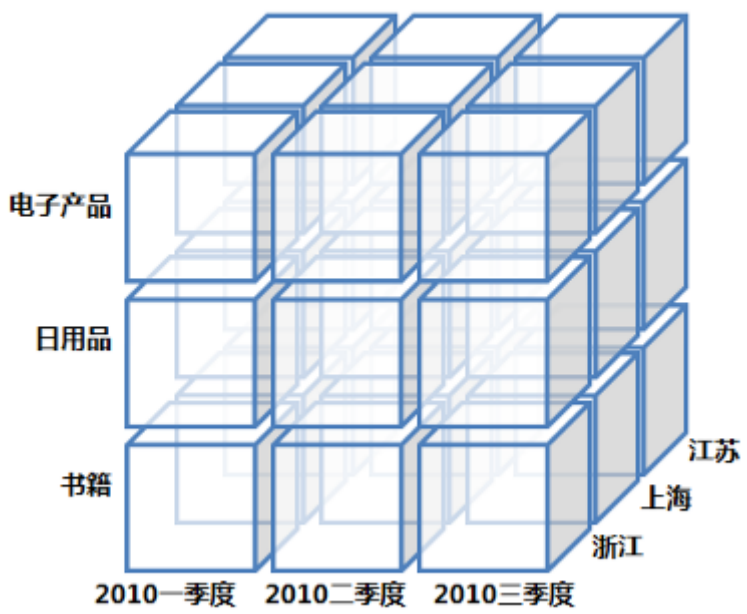
```
冗余：
    雪花模型符合业务逻辑设计，采用3NF设计，有效降低数据冗余；
    星型模型的维度表设计不符合3NF，反规范化，维度表之间不会直接相关，牺牲部分存储空间。

性能：
    雪花模型由于存在维度间的关联，采用3NF降低冗余，通常在使用过程中，需要连接更多的维度表，导致性能偏低；
    星型模型反三范式，采用降维的操作将维度整合，以存储空间为代价有效降低维度表连接数，性能较雪花模型高。

ETL处理：
    雪花模型符合业务ER模型设计原则，在ETL过程中相对简单，但是由于附属模型的限制，ETL任务并行化较低；
    星型模型在设计维度表时反范式设计，所以在ETL过程中整合业务数据到维度表有一定难度
```

下面的案例，对于商品再拆分出厂家、品类；用户的常住地再拆分出区域维度，改造为雪花模型。

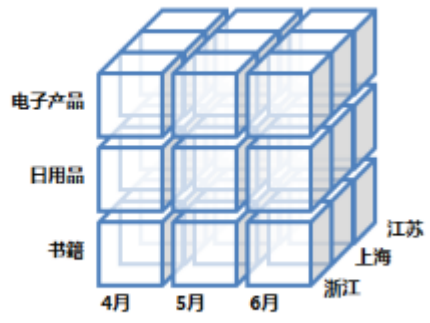
3.3.13. Cube 模型

CUBE模型其实就是多维模型的存储结构。我们可以通过季度、地区和品类三个维度，去看订单量、订单金额的统计结果。减少一个维度，就是上卷，增加一个维度就是下钻。就这么简单。

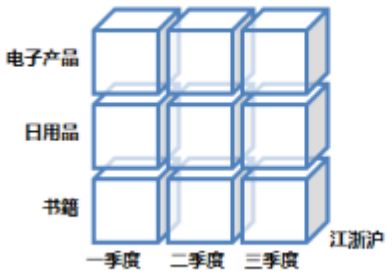
Kylin 会把每个组合方式计算一遍，然后存储下来。在查询的时候，Kylin 直接读取预计算的结果就好了，所以速度非常快，但是非常占存储。为了减少存储，Kylin 提供了剪枝的功能，就是把不常要的某个分支给删掉。



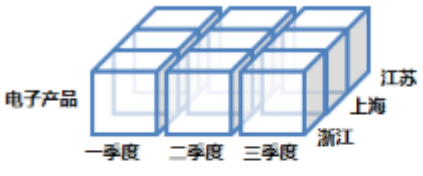
注意 cube 模型的一些基本操作：钻取（Drill-down）、上卷（Roll-up）、切片（Slice）、切块（Dice）以及旋转（Pivot）



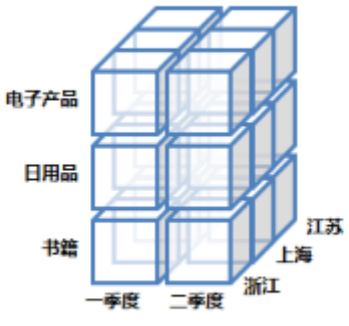
钻取(Drill-down)



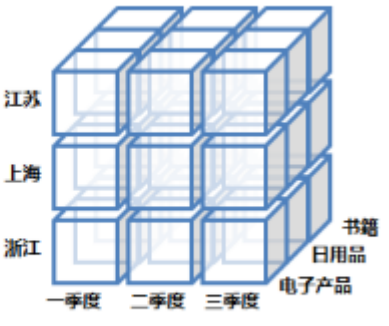
上卷(Roll-up)



切片(Slice)



切块(Dice)



旋转(Pivot)

3.3.14. 建模案例

电商平台（比如转转），经常需要对订单进行分析，以购物订单为例，以维度建模的方式设计该模型。

涉及到事实表为订单表、订单明细表，维度包括商品维度、用户维度、商家维度、区域维度、时间（日期）维度

事实表中的维度：

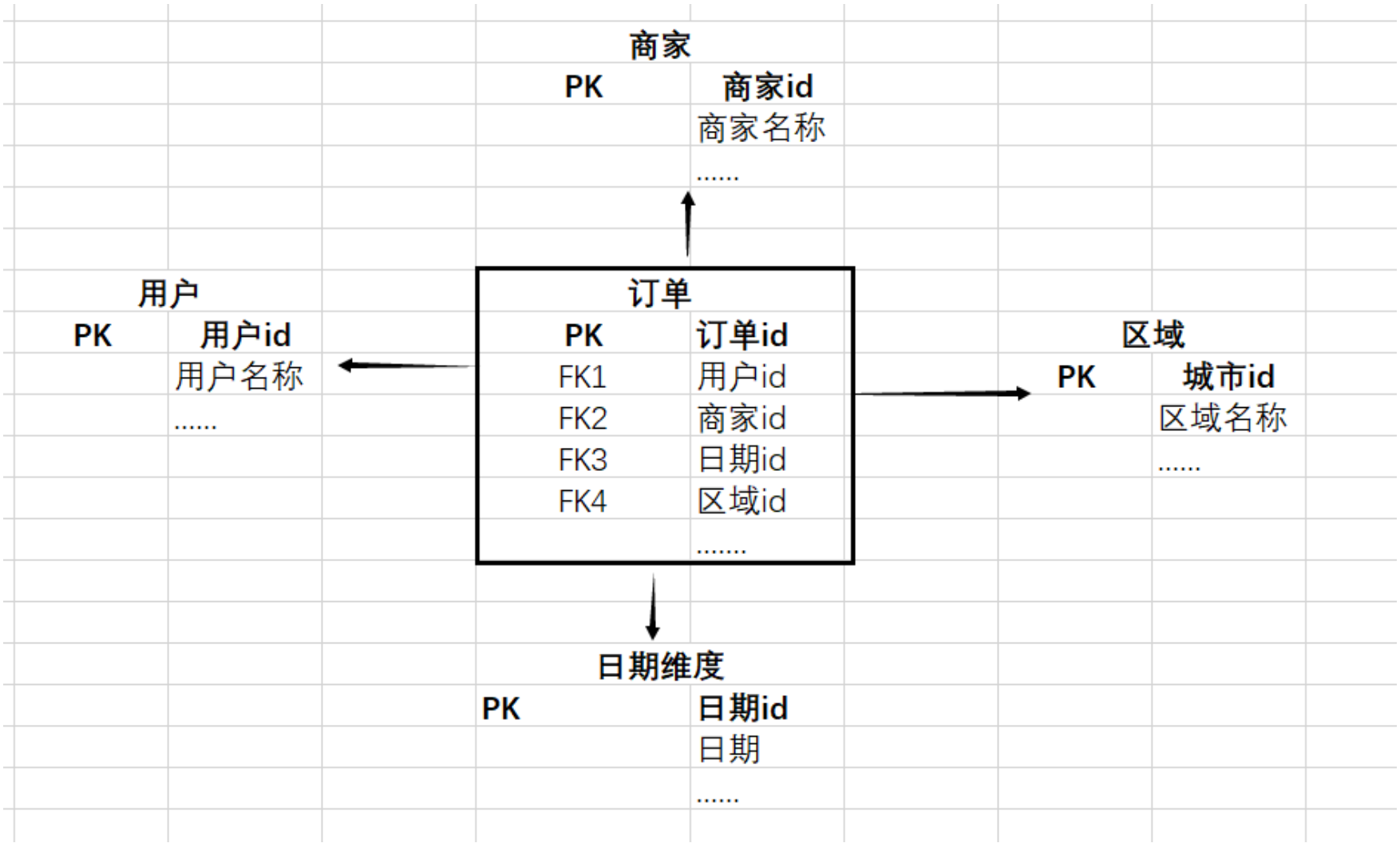
```
商品维度：商品ID、商品名称、商品种类、单价、来源地等
用户维度：用户ID、姓名、性别、年龄、常住地、职业、学历等
时间（日期）维度：日期ID、日期、周几、上/中/下旬、是否周末、是否假期、特殊日期等
优惠券：券ID、券类别/优惠方式、优惠金额
```

订单中包含的度量：商品件数、总金额、总减免

```
描述性属性：下单时间、支付时间、订单状态等
```

订单明细包含度量：商品ID、件数、单价、减免金额

```
描述性属性：加入购物车时间、状态
```



根据上图的设计可以发现，其实就是一种典型的星型模型，如果区域维度，拆解成多个渐变维度，则相当于去除了冗余，则变成雪花模型。

3.3.15. 忠告

在互联网领域，数仓模型的设计更关注灵活、快速响应和应对多变的市场环境，更加以快速解决业务、运营问题为导向，快速数据接入、快速业务接入，不存在一劳永逸。

3.4. 用户行为分析离线数仓案例

3.4.1. 项目背景

当今社会有很多的电商网站，这就存在着一些竞争关系，为了更好的设计一个网站，让一个电商网站浏览的人数更多，从而增加点击量和订阅量的，这样就需要我们对这个电商网站进行分析和数据挖掘，我们可以**根据每天浏览某电商网站的人数和访客量来判断一个网站的好坏和受欢迎程度，同时也可以根据外链的跳转率和访客或会员所用的浏览器等工具的分析来进行精准的广告推广，我们也可以根据地区的点击量和访客或是会员访问的时间的分析来进行合理的商品推广，精准的推荐**等操作。

网站分析（Web Analytics）主要指的是基于网站的用户浏览行为，即指用户访问网站时的所有访问、浏览、点击行为数据。比如点击了哪一个链接，在哪个网页停留时间最多，采用了哪个搜索项、总体会话时间等。而所有这些信息都可被保存在网站日志中。通过分析这些数据，可以获知许多对网站运营至关重要的信息。采集的数据越全面，分析就能越精准。对网站的点击流数据和运营数据进行分析，以监控网站的运营状况，为网站的优化提供决策依据。

总之，一个电商网站就应该设计出一款产品能让用户的体验好，能让用户精准的寻找想要购买的商品，能提高用户的转化率，能提广告的转化率。

3.4.2. 项目意义

网站流量统计分析，可以帮助网站管理员、运营人员、推广人员等实时获取网站流量信息，并从流量来源、网站内容、网站访客特性等多方面提供网站分析的数据依据。从而帮助提高网站流量，提升网站用户体验，让访客更多的沉淀下来变成会员或客户，通过更少的投入获取最大化的收入。

网站的眼睛	网站的大脑	网站的神经
访问者来自哪里? 访问者在寻找什么? 哪些页面最受欢迎? 访问者从哪里进入? 访问者从哪里跳出?	如何分解目标? 如何分配广告预算? 如何衡量产品表现? 哪些产品需要优化? 哪些指标需要关注?	网页布局合理吗? 网站导航清晰吗? 哪些功能存在问题 网站内容有效吗 转化路径靠谱吗?

点击流分析的意义可分为两大方面：技术上，可以合理修改网站结构及适度分配资源，构建后台服务器群组，业务上优化页面及业务流程设计，帮助企业更好地根据客户的兴趣来安排内容。终极目标是：**改善网站(电商、社交、电影、小说)的运营，获取更高投资回报率（ROI）**

3.4.3. 点击流数据模型

日志的生成渠道

```
1、是网站的web服务器软件（apache、nginx、tomcat）所记录的web访问日志；
2、是通过在页面嵌入自定义的JS代码来获取用户的所有访问行为（比如鼠标悬停的位置，点击的页面组件等），然后通过AJAX请求到后台记录日志；这种方式所能采集的信息最全面；
3、通过在页面上埋点1像素的图片，将相关页面访问信息请求到后台记录日志；
```

日志数据内容详述

```
在实际操作中，有以下几个方面的数据可以被采集：
1、访客的系统属性特征。比如所采用的操作系统、浏览器、域名和访问速度等。
2、访问特征。包括停留时间、点击的URL、所点击的“页面标签<a>”及标签的一些属性（比如业务entity<商品、电影、歌曲、小说名称>的名称）等。
3、来源特征。包括来访URL，来访IP等。
4、产品特征。包括所访问的产品编号、产品类别、产品颜色、产品价格、产品利润、产品数量和特价等级等。
```

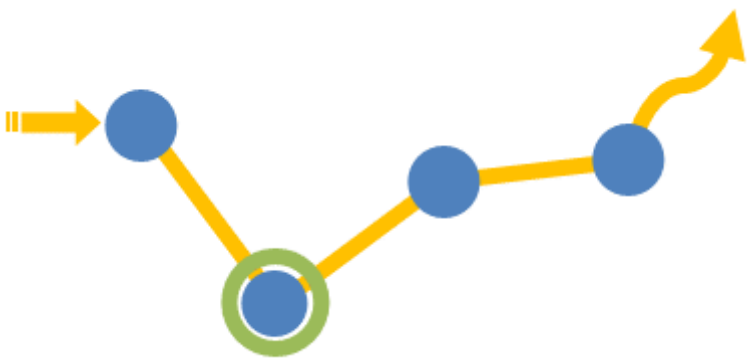
日志数据案例

```
GET /log.gif?t=item.010001&m=UA-J2011-1&pin=-&uid=1679790178&sid=1679790178|12&v=je=1$sc=24-bit$sr=1600x900$ul=zh-cn$cs=GBK$dt=xxxx$hn=item.jd.com$fl=16.0r0$os=win$br=chrome$bv=39.0.2171.95$wb=1437269412$xb=1449548587$yb=1456186252$zb=12$cb=4$usc=direct$ucp=-$umd=none$uct=-$ct=1456186505411$lt=0$tad=-$sku=1326523$cid1=1316$cid2=1384$cid3=1405$brand=20583$pinid=-&ref=&rm=1456186505411 HTTP/1.1
```

```
关键词：%E6%95%B0%E6%8D%AE%E4%BB%93%E5%BA%93hive = 数据仓库hive
https://www.baidu.com/s?wd=%E6%95%B0%E6%8D%AE%E4%BB%93%E5%BA%93hive&rsv_spt=1&rsv_iqid=0xe24eca32000e6c54&issp=1&f=8&rsv_bp=1&rsv_idx=2&ie=utf-8&tn=baiduhome_pg&rsv_enter=1&rsv_dl=tb&rsv_sug3=20&rsv_sug1=16&rsv_sug7=100&rsv_sug2=0&rsv_btype=i&inputT=6473&rsv_sug4=6695
```

3.4.4. 点击流

点击流这个概念更注重用户浏览网站的整个流程，网站日志中记录的用户点击就像是图上的"点"，而点击流更像是将这些"点"串起来形成的"线"。也可以把"点"认为是网站的Page，而"线"则是访问网站的Session。所以点击流数据是由网站日志中整理得到的，它可以比网站日志包含更多的信息，从而使基于点击流数据统计得到的结果更加丰富和高效。



点击流数据在具体操作上是由散点状的点击日志数据梳理所得，从而，点击数据在数据建模时应该存在两张模型表（pageviews 和 visits）：

1、用于生成点击流的**原始访问日志表**

时间戳	IP地址	请求URL	Referal	响应吗	流量	br信息
2012-01-01 12:31:12	101.0.0.1	/a/...	somesite.com			
2012-01-01 12:31:16	201.0.0.2	/a/...	https://naixuejiaoyu.com/			
2012-01-01 12:33:06	201.0.0.2	/b/...	/a/...			
2012-01-01 15:16:39	234.0.0.3	/c/...	google.com			
2012-01-01 15:17:11	101.0.0.1	/d/...	/c/...			
2012-01-01 12:39:23	201.0.0.2	/e/...	/b/....			

2、页面点击流模型 pageviews 表

Session	IP地址	时间	访问页面URL	停留时长	第几步
S001	101.0.0.1	2012-01-01 12:31:12	/a/....	30	1
S002	201.0.0.2	2012-01-01 12:31:16	/a/....	110	1
S002	201.0.0.2	2012-01-01 12:33:06	/b/....	317	2
S002	201.0.0.2	2012-01-01 13:39:23	/e/....	30	3
S003	101.0.0.1	2012-01-01 15:17:11	/d/....	30	1
S004	234.0.0.3	2012-01-01 15:16:39	/c/....	30	1

3、点击流模型visits表(按session聚集的页面访问信息)

Session	起始时间	结束时间	进入页面	离开页面	访问页面数	IP	cookie	referal
S001	2012-01-01 12:31:12	2012-01-01 12:31:12	/a/...	/a/...	1	101.0.0.1	User01	somesite.com
S002	2012-01-01 12:31:16	2012-01-01 12:39:23	/a/...	/e/...	3	201.0.0.2	User02	google.com
S003	2012-01-01 12:35:42	2012-01-01 12:35:42	/d/...	/d/...	1	101.0.0.1	User03	baidu.com
S004	2012-01-01 15:16:39	2012-01-01 15:16:39	/c/...	/c/...	1	234.0.0.3	User01	google.com
.....	.....	.....	.....	.....	.....	.....	.....	.....

这就是点击流模型。当 WEB 日志转化成点击流数据的时候，很多网站分析度量的计算变得简单了，这就是点击流的“魔力”所在。**基于点击流数据我们可以统计出许多常见的网站分析度量指标**

3.4.5. 点击流常见分析指标

```
PV
UV
VV
独立IP
停留时长
访问深度
转化率
跳出率
退出率
```

3.4.6. 经典分析需求实现

```
1、基础流量分析，比如一些非常核心的指标：PV，IP，UV，独立访客，还可以做趋势，明细，对比等分析
2、来源分析，包括来源分类，来源页面，搜索引擎，搜索词等
3、受访分析，包括访问页面，停留时间，访问深度，跳出率，退出率等
4、访客分析，包括终端详情，新老访客，忠诚度，活跃度，用户粘性等
5、转化路径分析，或者叫漏斗分析，如注册或下载，下单等
6、用户分析，包括访客，会员，会话等分析
7、热力图分析
8、用户轨迹分析
9、ROI分析
10、RFM分析
......
```

友盟，易分析，.....

3.4.7. 数仓技术架构

整体流程基本上就是依据数据的处理流程进行，依此有以下几个大的步骤：

- 数据采集：首先，通过页面嵌入 JS 代码的方式获取用户访问行为，并发送到 web 服务的后台记录日志，然后，将各服务器上生成的点击流日志通过实时或批量的方式汇聚到 HDFS 文件系统中。当然，一个综合分析系统，数据源可能不仅包含点击流数据，还有数据库中的业务数据（如用户信息、商品信息、订单信息等）及对分析有益的外部数据。
- 数据预处理：通过 MapReduce，HiveQL，Kettle，Spark 等程序对采集到的点击流数据进行预处理，比如清洗，格式整理，滤除脏数据等
- 数据入库建模：将预处理之后的数据导入到 HIVE 仓库中相应的库和表中
- 数据分析：项目的核心内容，即根据需求开发 HiveSQL 分析语句，得出各种指标统计结果
- 数据展现：将分析所得数据进行可视化，可提供资助可视化分析平台

3.5. 未来数仓发展趋势

3.5.1. 自研平台

自研集资源管理，权限管理，任务管理，运行调度，依赖处理，自助分析，跟踪监控，抽取推送等多个功能于一体的工具平台。

3.5.2. 关注数据质量

关注数据的一致性，完整性，合理性，及时性等

3.5.3. 数据血缘分析

包括异常分析，产出分析，依赖分析等

3.5.4. 开放查询平台

包括数据自助查询，数据权限管理，数据资源管理，查询历史管理，查询分析，数据使用记录等功能

3.5.5. 数据地图

包括元数据治理，指标体系建设，数据生命周期，数据资产，数据地图等功能

4. 本次内容总结

今天的课程是 全域离线数仓的 第一次。第一次课程的内容，旨在帮助大家理解，为什么要构建数仓，构建数仓的流程，怎么构建数仓等等相关问题。

```
1、认识数据仓库
2、数仓构建流程
3、数仓建模理论
4、用户行为分析离线数仓案例
5、未来数据仓库发展
```

今天的课程的最大目的，是希望大家对于构建企业级数仓有一个全面的体系化认知。

5. 本次课程作业

5.1. 数据格式

原始数据 access.log

样例：

```
58.215.204.118 - - [18/Sep/2013:06:51:35 +0000] "GET /wp-includes/js/jquery/jquery.js?ver=1.10.2 HTTP/1.1" 304 0
"http://blog.fens.me/nodejs-socketio-chat/" "Mozilla/5.0 (Windows NT 5.1; rv:23.0) Gecko/20100101 Firefox/23.0"
```

字段解释：

```
1、访客 ip 地址： 58.215.204.118
2、访客用户信息： - -
3、请求时间： [18/Sep/2013:06:51:35 +0000]
4、请求方式： GET
5、请求的 url： /wp-includes/js/jquery/jquery.js?ver=1.10.2
6、请求所用协议： HTTP/1.1
7、响应码： 304
8、返回的数据流量： 0
9、访客的来源 url： http://blog.fens.me/nodejs-socketio-chat/
10、访客所用浏览器： Mozilla/5.0 (Windows NT 5.1; rv:23.0) Gecko/20100101 Firefox/23.0
```

5.2. 数据 ETL

清洗上述数据，得到规整的结果：

时间戳	IP地址	请求URL	Referal	浏览器属性	……
2012-01-01 12:31:12	101.0.0.1	/a/…	somesite.com		
2012-01-01 12:31:16	201.0.0.2	/a/…	aura.cn		
2012-01-01 12:33:06	101.0.0.2	/b/…	baidu.com		
2012-01-01 15:16:39	234.0.0.3	/c/…	google.com		
2012-01-01 15:17:11	101.0.0.1	/d/…	/c/…		
2012-01-01 15:19:23	101.0.0.1	/e/…	/d/….		

5.3. 需求实现

从上述清洗结果中梳理出以下两种模型数据："**页面点击流模型 Pageviews 表**" 和 "**点击流模型 Visits 表**"

页面点击流模型 Pageviews 表

SessionID	userid	时间	访问页面URL	页面停留时长	第几步
S001	User01	2012-01-01 12:31:12	/a/….	30	1
S002	User02	2012-01-01 12:31:16	/a/….	110	1
S002	User02	2012-01-01 12:33:06	/b/….	120	2
S002	User02	2012-01-01 12:35:06	/e/….	30	3

点击流模型 Visits 表

Session	起始时间	结束时间	进入页面	离开页面	访问页面数	IP	cookie	referal
S001	2012-01-01 12:31:12	2012-01-01 12:31:12	/a/…	/a/…	1	101.0.0.1	User01	somesite.com
S002	2012-01-01 12:31:16	2012-01-01 12:35:36	/a/…	/e/…	3	201.0.0.2	User02	naixue.cn
S003	2012-01-01 12:35:42	2012-01-01 12:35:42	/c/…	/c/…	1	234.0.0.3	User03	baidu.com
S003	2012-01-01 15:16:39	2012-01-01 15:19:23	/c/…	/e/…	3	101.0.0.1	User01	google.com
……	……	……	……	……	……	……	……	……